NICER

DTS TESTING PLAN FOR NICER MISSION
(adapted from Astro-H)

Version 1.0

DATE Nov 2016

GSFC
Greenbelt, Maryland

Prepared by: L Angelini and J Eggen (HEASARC)

Table of Contents

.

CHANGE RECORD PAGE (1 of 2)

DOCUMENT TITLE :Interface Control Document HEASARC-Astro-E2 GOF			
ISSUE	DATE	PAGES AFFECTED	DESCRIPTION
Version 1.0	Nov 2016	All	Version 1

1 Introduction

This document describes the planned tests for exercising the DTS protocol, which transfers data among separate data centers. This is specific to the data transfer between the Nicer processing site and the HEASARC archive. It assumes that all sites participating in the testing have already set up a machine with suitable disk space. General requirements for the machine hardware and software as well information and code for DTS are available at the following web address:

http://heasarc.gsfc.nasa.gov/dts/dts.html

The current DTS version is 6.3.1. This version is installed in the NuStar, Swift, Suzaku and Hitomi accounts at Goddard in the archive and/or processing machine. The test commands should be verified by sending email to the following people:

Stephanie.G.Zonak@nasa.gov
joe.eggan@nasa.gov
craig.b.markwardt@nasa.gov
lorella.angelini-1@nasa.gov

2 Test Summary

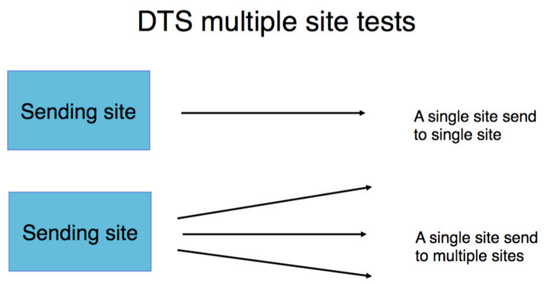
This test document includes two types of tests one to establish that dts functionality and another to mimic the exact commands to use during the operations.

The functionality test consists in four types of different DTS commands to verify the operation of the DTS software across the Nicer sites.

- The first consists of single site to single site transfers, e.g. each site transfers data to another site.

- This is to ensure that all sites can talk to each other.

- The second consists of single site to multiple sites transfers, e.g. one site will transfer data to all other sites that join the test.


DTS multiple site tests
Sending site
A single site send to single site
Sending site
A single site send to multiple sites

.

- The third is a stress test to determine whether the DTS can handle simultaneous processing of many retrievals. This is achieved with one site sending a large number of transfers in a small amount of time.

- The last is a set of failure/recovery tests, used to determine whether the DTS properly handles transfer problems. The recovery process is tested on recoverable errors, while unrecoverable errors are tested by ensuring proper notification of the problem is sent to the DTS operator.

All tests, described in detail in the following sections, are to be performed in combination with every other participating site.

The additional test instead uses the exact commands used in during the operation and send the data to the archive after the processing is completed.

2.1 General commands/procedures

A) SEND OPTIONS. The behavior of the send option can be modified with several flags. All options are tested. These are:

- Option -f . Used to send files listed in a text file (either flat or with directory structure):

dts -se SITE -f filelist.txt

Where filelist.txt contains the following files
example.gif
example.evt
-OR-
obsid/inst1/screened/example.evt
obsid/inst1/images/example.gif

- Option -l. Used to send files listed on the command line (either flat or w/ directory structure). For example,

*dts -se SITE -l *.**
-OR-
dts -se SITE -l `find obsid -type f -print`

- Option z. Compresses files with gzip in ftp directory before sending
  - *dts -se SITE -z ALL -f filelist.txt*

- Option -split. Splits files into smaller pieces in ftp directory before sending
  - *dts -se SITE -split -f filelist.txt*

- Option -t. If AUTORUN is set to true in the dts.config file, TYPE will spawn matching script in dts.scripts
  - *dts -se SITE -t TYPE -f filelist.txt*

.

- Send to multiple sites. The send command can be used to send data to multiple sites simultaneously by separating them with commas. All previous options may also be used.
    - *dts -se SITE1,SITE2,SITE3 -f filelist.txt*

B) AUTOCLEAN. If set to TRUE in dts.config, this option cleans out the ftp area of the sending site after the receiving site sends an acknowledgement email.

C) ONE TIME PASSWORD. To increase security a one time password mechanism has been implemented within DTS. To use the one-time password option to access FTP site, the key USEOTP in the configuration file (dts.config) needs to be set. However this was implemented before the 'sftp' protocol was available. If the machine where DTS is run has installed 'sftp' the one time password feature is unnecessary. For our tests USEOTP is set to FALSE in the config file, and USESFTP (also in the config file) is set to TRUE.

D) RECOVERY MODE. There are potential cases where the transfer is interrupted but can be recovered:

- Sending site's FTP server is temporarily down. [To simulate : Change to invalid ftp password or site in dts.sitelist]
- Receiving site's staging directory is temporarily unavailable or out of disk space. [To simulate: Change DTS_STG to invalid directory in dts.config]. To recover the transfer, the following type of DTS command is used :

dts -r dts######_######G.mail

E) FAILURE NOTIFICATION. In unrecoverable failure scenarios the DTS operator must be notified so that appropriate action can be taken. The following failures fit into this category:

- Sending site's FTP directory is unavailable or out of space. The DTS send will fail before sending any message. [To simulate: Change DTS_FTP in dts.config to invalid directory] This assumes that the send operation is occurring manually or the script that spawns dts can identify this failure.
- Files are deleted before retrieval, i.e. directory deleted from ftp area before informing receiving site. Email should be sent to DTS operator.
- Files are corrupted in transfer. Email should be sent to the DTS operator. [To simulate: Corrupt some files in FTP area, echo blah >> filename]

3 Functionality Test Detail

Each test uses the following general steps:

```
FROM SITE1 send to SITE2: dts -se SITE2 [send options]
SITE2 RECEIVING MODE    : dts -g
SITE1 CLEANING MODE     : dts -g   (AUTOCLEAN=TRUE)
```

Although the DTS is largely hands-off once in operation, during the testing period it must be monitored closely to ensure that it is operating properly. Each site must have a contact person to coordinate tests between sites. After each DTS send, an email must be sent to the the receiving site's contact person so that they can verify that the SEND message has arrived and can then run ``dts -g" The email message must contain the identifier for the test that is being run, listed as "Test: TEST_NAME" in the procedures below. Once the files have been retrieved, the receiving site must verify that they have arrived intact by comparing them to the testbed. Then, they email the sending site's contact person about the results of the test. The sending site then runs ``dts -g" and verifies that the ftp area has been cleaned (AUTOCLEAN is TRUE for all tests).

To diagnostic problems with a DTS test or verify its proper operation it is necessary to know your way around the DTS. The central location for all DTS parameters is the dts.config file. The following parameters defined in dts.config are important for evaluating the test operations:

- DTS_IN - Location of logs when you are the receiving site
- DTS_OUT - Location of logs when you are the sending site
- DTS_STG - Location of incoming files, found in time-coded directories
- DTS_FTP - Location of outgoing files, found in time-coded directories
- MAILFILE - Location of mail file (In some cases it may be useful to view the contents of the mail file before running ``dts -g")
- SPLITPATH - Location of script used to split large data files, and size at which to split them.

After all testing is completed, the staging areas may be cleaned out to save disk space, however, we request that you retain the contents of the log directories, as they contain valuable information on transfer rate, etc.

3.1 Setup

The designation for the Nicer processing site and the HEASARC archive site machines are :

```
SITE ID:  MAIL ACCOUNT:                         FTP ACCOUNT:

NICERMOC ??dtsmail@???dc.gsfc.nasa.gov ??dtsftp@??dc.gsfc.nasa.gov
NICERHEA   nidtsmail@ headts.gsfc.nasa.gov nidtsftp@ headts.gsfc.nasa.gov
```

i.e., NICERHEA is Goddard's archiving machine, and NICERMOC is Goddard's processing machine. All sites regardless of whether they send, receive, or both must retrieve the testbed from the website:

ftp://heasarc.gsfc.nasa.gov/nicer/prelaunch/dtstest/dtstest.tar.gz
ftp://heasarc.gsfc.nasa.gov/nicer/prelaunch/dtstest/validated.tar.gz
ftp://heasarc.gsfc.nasa.gov/nicer/prelaunch/dtstest/notvalidatedtar.gz

Make a */testbed* directory, and unpack the tar files into that directory:

gunzip -c yyyyyyyy.tar.gz | tar xvf -

These are files are used in the following way:

.

- Content of dtstest is the standard dts test and exercise the command under section 3.2, 3.3, 3.4 and 3.5. The dtstest.tar creates under the */testbed* directory the test data and scripts for use with dts and the testing procedure. The content is the following :

  ```
  ad71031000.002/
  doc/
  filelist.txt
  flat/
  scripts/
  ```

- Content of notvalidated.tar validated.tar and ni_test_files.tar exercise the test under section 4 and are specific for NICER. The content of the tar files is described in section 4.

Note that the test procedures are written primarily from the perspective of the sending site, however, special notes to the receiving site are preceded by ``Receiving site:''. The following tests are to exercise only single site to single site data transfers since for Nicer mission the data are not transferred to multiple sites.

3.2 Single Site to Single NICERMOC to NICERHEA

This section concerns data transfer from NICERMOC to NICERHEA and are designed to test the various DTS options.

A) Test: SIMPLE_FLAT (NICERMOC to NICERHEA)

Change to the `flat' directory in your testbed. NOTE: at this point dts should be in your path and the DTS_CONFIG environmental variable should be set. Edit the VERSION file in this directory to reflect your site id (e.g. NICERMOC or NICERHEA) and the test name (i.e. SIMPLE_FLAT). From NICERMOC :

*dts -se NICERHEA -l **

This sends the VERSION, p0007422101m1s001mievli0000.fit and rp200008.evt files to NICERHEA. Inform the receiving site.

Receiving site (NICERHEA): After running ``dts -g'' change to the staging area (DTS_STG in dts.config) and go into the most recent directory. Verify that the VERSION file reflects the expected site and test name. Use diff or the ftool fdiff to compare the received files with their counterparts in the local testbed. NOTE: Due to forking operations, the ``dts -g'' command will not print the shell prompt after it completes. After the DTS messages are done printing, press Enter to return the prompt.

B) Test: COMPRESS_FLAT (NICERMOC to NICERHEA)

Edit the VERSION file to reflect the test name. From the NICERMOC testbed's `flat' directory:

.

*dts -se NICERHEA -z ALL -l **

This sends the VERSION, p0007422101m1s001mievli0000.fit and rp200008.evt files to NICERHEA, compressing them with gzip in the process. Inform the receiving site.

Receiving site (NICERHEA): Verify the VERSION file. Either uncompress the files and then diff or use the ftool fdiff which uncompresses on-the-fly to compare the received files with their counterparts in the local testbed.

C) Test: SPLIT_FLAT (NICERMOC to NICERHEA)

For this test we will split the files into 200 kilobyte pieces before transfer. The split path should be defined in dts.config as follows:

```
SPLITPATH="/usr/bin/split -b 200k"
```

(Note: The small size split is for testing purposes only; remember to change this back to 100000k when this test is complete.) The local split command may be located elsewhere. Modify the path to reflect it. Edit the VERSION file to reflect the test name. From the NICERMOC testbed's `flat' directory:

*dts -se NICERHEA -split -l **

This sends the p0007422101m1s001mievli0000.fit and rp200008.evt files to NICERHEA, splitting them into smaller pieces in the process.

Receiving site (NICERHEA): Verify the VERSION file. The files should be reconstituted in the staging area. Use diff or fdiff to compare the received files with their counterparts in the local testbed.

D) Test: SCRIPT_PATH (NICERMOC to NICERHEA)

Change to the top level directory of the testbed. Edit the ad71031000.002/VERSION file to reflect your site id and the test name. From NICERMOC:

dts -se NICERHEA -type PTEST -f filelist.txt

Receiving site (NICERHEA): This test takes more preparation than prior tests. The transmission of the directory structure must be verified through the use of a simple script. Edit dts.scripts to associate PTEST with the full path to scripts/dirlist.pl in the testbed. Edit the first line of dirlist.pl to have the path to perl on your system. Run dirlist.pl manually with no arguments before running DTS. The following should be printed:

```
AUTORUN MESSAGES: dirlist.pl
ERROR: No files
```

.

If it is not already, set AUTORUN to TRUE in dts.config. Now, run ``dts -g" The script should create a text file named dirstruct.txt, which contains the directory structure of the original files,inside the staging directory with the transferred files. In order to compare the received files with the testbed, it is necessary to restore the directory structure. From within the directory that contains the retrieved files, run the dirbuild.pl script from the testbed, giving dirstruct.txt as the argument. For example, if the testbed directory is in the dtsops home directory:

~/testbed/scripts/dirbuild.pl dirstruct.txt

This reconstructs the directory structure and allow comparison with the testbed using the "diff -r" command. Note: All remaining tests use this procedure to verify the data integrity of the transferred files.

3.3 Single Site to Single NICERHEA to NICERMOC

This section concerns data transfer from NICERHEA to NICERMOC and are designed to test the various DTS options. It is not expected that NICERHEA sends data to NICERMOC but the tests are to exercise the validity of the DTS installation and options.

A) Test: SIMPLE_FLAT (NICERHEA to NICERMOC)

Change to the `flat' directory in your testbed. NOTE: at this point dts should be in your path and the DTS_CONFIG environmental variable should be set. Edit the VERSION file in this directory to reflect your site id (e.g. NICERMOC or NICERHEA) and the test name (i.e. SIMPLE_FLAT). From NICERHEA :

*dts -se NICERMOC -l **

This should send the VERSION, p0007422101m1s001mievli0000.fit and rp200008.evt files to NICERMOC. Inform the receiving site.

Receiving site (NICERMOC): After running ``dts -g" change to the staging area (DTS_STG in dts.config) and go into the most recent directory. Verify that the VERSION file reflects the expected site and test name. Use diff or the ftool fdiff to compare the received files with their counterparts in the local testbed. NOTE: Due to forking operations, the ``dts -g" command will not print the shell prompt after it completes. After the DTS messages are done printing, press Enter to return the prompt.

B) Test: COMPRESS_FLAT (NICERHEA to NICERMOC)

Edit the VERSION file to reflect the test name. From the NICERHEA testbed's `flat' directory:

*dts -se NICERMOC -z ALL -l **

This sends the VERSION, p0007422101m1s001mievli0000.fit and rp200008.evt files to NICERMOC, compressing them with gzip in the process. Inform the receiving site.

.

Receiving site (NICERMOC): Verify the VERSION file. Either uncompress the files and then diff or use the ftool fdiff which uncompresses on-the-fly to compare the received files with their counterparts in the local testbed.

C) Test: SPLIT_FLAT (NICERHEA to NICERMOC)

For this test we will split the files into 200 kilobyte pieces before transfer. The split path should be defined in dts.config as follows:

```
SPLITPATH="/usr/bin/split -b 200k"
```

(Note: The small size split is for testing purposes only; remember to change this back to 100000k when this test is complete.) The local split command may be located elsewhere. Modify the path to reflect it. Edit the VERSION file to reflect the test name. From the NICERHEA testbed's `flat' directory:

*dts -se NICERMOC -split -l **

This sends the p0007422101m1s001mievli0000.fit and rp200008.evt files to NICERMOC, splitting them into smaller pieces in the process.

Receiving site (NICERMOC): Verify the VERSION file. The files should be reconstituted in the staging area. Use diff or fdiff to compare the received files with their counterparts in the local testbed.

D) Test: SCRIPT_PATH (NICERHEA to NICERMOC)

Change to the top level directory of the testbed. Edit the ad71031000.002/VERSION file to reflect your site id and the test name. From NICERHEA:

dts -se NICERMOC -type PTEST -f filelist.txt

Receiving site (NICERMOC): This test takes more preparation than prior tests. The transmission of the directory structure must be verified through the use of a simple script. Edit dts.scripts to associate PTEST with the full path to scripts/dirlist.pl in the testbed. Edit the first line of dirlist.pl to have the path to perl on your system. Run dirlist.pl manually with no arguments before running DTS. The following should be printed:

```
AUTORUN MESSAGES: dirlist.pl
ERROR: No files
```

If it is not already, set AUTORUN to TRUE in dts.config. Now, run ``dts -g" The script should create a text file named dirstruct.txt, which contains the directory structure of the original files,inside the staging directory with the transferred files. In order to compare the received files with the testbed, it is necessary to restore the directory structure. From within the directory that contains the retrieved files, run the dirbuild.pl script from the testbed, giving dirstruct.txt as the argument. For example, if the testbed directory is in the dtsops home directory:

.

~/testbed/scripts/dirbuild.pl dirstruct.txt

This reconstructs the directory structure and allow comparison with the testbed using the "diff -r" command. Note: All remaining tests use this procedure to verify the data integrity of the transferred files.

3.4 Stress Test

A) Test: STRESS (NICERMOC to NICERHEA)

The purpose of this test to stress the DTS by requiring it to have many more transfers waiting in the mailbox than it can run at one time. This exercises the queueing process which allows up to 5 processes (the default value for CONCUR_MAX) to run at one time. Therefore, it is choosen a value larger than CONCUR_MAX. For this test 2*CONCUR_MAX = 10.
A simple script has been provided to run this test. From the NICERMOC testbed directory:

scripts/stress.pl NICERHEA
After the script completes, running DTS send 10 times, inform the receiving site.

Receiving site (NICERHEA): After running ``dts -g" monitor the dts process with the top command. You should see at most 5 such processes at any one time. After all have completed, compare the received files with the testbed (see SCRIPT_PATH).

As a shortcut, compare all directories associated with the stress test to the most recent directory. They should all match except for the VERSION file. Then, reconstruct the directory structure of the one all were compared to, and compare the the reconstructed directory to the testbed. If there is some confusion as to which directories are associated with the stress test, use "grep STRESS */VERSION" from the staging directory.

B) Test: STRESS (NICERHEA to NICERMOC)

The purpose of this test to stress the DTS by requiring it to have many more transfers waiting in the mailbox than it can run at one time. This exercises the queueing process which allows up to 5 processes (the default value for CONCUR_MAX) to run at one time. Therefore, it is choosen a value larger than CONCUR_MAX. For this test 2*CONCUR_MAX = 10.
A simple script has been provided to run this test. From the NICERMOC testbed directory:

scripts/stress.pl NICERMOC
After the script completes, running DTS send 10 times, inform the receiving site.

Receiving site (NICERMOC): After running ``dts -g" monitor the dts process with the top command. You should see at most 5 such processes at any one time. After all have completed, compare the received files with the testbed (see SCRIPT_PATH).

.

As a shortcut, compare all directories associated with the stress test to the most recent directory. They should all match except for the VERSION file. Then, reconstruct the directory structure of the one all were compared to, and compare the the reconstructed directory to the testbed. If there is some confusion as to which directories are associated with the stress test, use "grep STRESS */VERSION" from the staging directory.

3.5 Failure/Recovery

A1) Test: RECO_BADPASS (NICERMOC to NICERHEA)

This test simulates the failure that would occur when a sending site's ftp site is temporarily unavailable. Edit the ad71031000.002/VERSION file to reflect the test name.
From the NICERMOC testbed directory:

dts -se NICERHEA -type PTEST -f filelist.txt

Inform the receiving site.
Receiving site (NICERHEA): Edit dts.sitelist, inserting an 'X' before the encoded password of the sending site. This renders the password invalid. Now, run ``dts -g''. It should fail and send and email to dtsops at your site. Edit dts.sitelist, removing the 'X'. In the log directory (DTS_IN in dts.config) run ``dts -r'' on the most recent file with ".mail" as its extension.

dts -r dts######_######G.mail

After completion, compare the received files with the testbed (see SCRIPT_PATH).

A2) Test: RECO_BADPASS (NICERHEA to NICERMOC)

This test simulates the failure that would occur when a sending site's ftp site is temporarily unavailable. Edit the ad71031000.002/VERSION file to reflect the test name.
From the NICERHEA testbed directory:

dts -se NICERMOC -type PTEST -f filelist.txt

Inform the receiving site.
Receiving site (NICERMOC): Edit dts.sitelist, inserting an 'X' before the encoded password of the sending site. This renders the password invalid. Now, run ``dts -g''. It should fail and send and email to dtsops at your site. Edit dts.sitelist, removing the 'X'. In the log directory (DTS\_IN in dts.config) run ``dts -r'' on the most recent file with ``.mail'' as its extension.

dts -r dts######_######G.mail

After completion, compare the received files with the testbed (see SCRIPT_PATH)\.

B1) Test: FAIL_CORRUPT (NICERMOC to NICERHEA)

This test simulates the failure when files are corrupted in transfer. Edit the ad71031000.002/VERSION file to reflect the test name.
From the NICERMOC testbed directory:

dts -se NICERHEA -type PTEST -f filelist.txt

Change to the directory in the ftp area that was just "sent." You may want to look at the VERSION file to be sure. Corrupt few files in that directory (e.g. `echo blah >> filename`). Inform the retrieving site.
Receiving site (NICERHEA): The ``dts -g" operation should detect an error and send email to the operator account associated with your DTS.

B2) Test: FAIL_CORRUPT (NICERHEA to NICERMOC)

This test simulates the failure when files are corrupted in transfer. Edit the ad71031000.002/VERSION file to reflect the test name.
From the NICERHEA testbed directory:

dts -se NICERMOC -type PTEST -f filelist.txt

Change to the directory in the ftp area that was just "sent." You may want to look at the VERSION file to be sure. Corrupt few files in that directory (e.g. `echo blah >> filename`). Inform the retrieving site.
Receiving site (NICERMOC): The ``dts -g" operation should detect an error and send email to the operator account associated with your DTS.

4 DTS Test to Mimic the NICER Label

The following labels corresponds to the the different type of data that the MOC send to HEASARC during operation. The labels are :

nidata1 – data not validated arrive at the archive encrypted
nihkgen1 trend data not validated arrive at the archive encrypted
nihkmpu1 – trend data not validated arrive at the archive encrypted
nihkconf1 – trend data not validated arrive at the archive encrypted

nidata2 – data validated arrive at the archive not encrypted
nihkgen2 – trend validated arrive at the archive not encrypted
nihkmpu2 – trend validated arrive at the archive not encrypted
nihkconf2 – trend validated arrive at the archive not encrypted
nidbtable – database for the validated data

.

4.1 NICER Tar File Contents

In order to mimic NICER data transfers, data has been packaged into two tar files - one with the validated data (``validated.tar.gz") and one with the not validated data (``notvalidated.tar.gz"). The sequence number used is 80906801X where X is either 0 or 1 to distinguish between validated and not validated data. The tar file structure is the following :

```
validated/
     nidata2/
         809068011/
         nidata2_filelist.txt
     nidbtable/
         nicermaster.tdat
         nidbtable_filelist.txt
     nihkconf2/
         ...
     nihkgen2/
         ...
     nihkmpu2/

 notvalidated/
     nidata1/
         809068011/
         nidata1_filelist.txt
     nihkconf1/
         ...
     nihkgen1/
         ...
     nihkmpu1/
       ...
```

File associated to the observations are listed in *nidata1_filelist.txt* and *nidata2_filelist.txt* . The files listed in *nidata2_filelist.txt* are not encrypted and do not have the suffix ‘gpg ‘ :

```
80906801X/auxil/ni80906801X.att.gz.gpg
80906801X/auxil/ni80906801X.cat.gz
80906801X/auxil/ni80906801X.hk.gz.gpg
80906801X/auxil/ni80906801X.mkf.gz.gpg
80906801X/auxil/ni80906801X.orb.gz.gpg

80906801X/log/ni80906801X_errlog.html.gz
80906801X/log/ni80906801X_flinfo.html.gz
80906801X/log/ni80906801X_hdpage.html.gz
80906801X/log/ni80906801X_index.html.gz
80906801X/log/ni80906801X_job.par.gz
```

.

```
80906801X/log/ni80906801X_joblog.html.gz
80906801X/log/ni80906801X_lv1.par.gz

80906801X/xti/event_cl/ni80906801X_0mpu7_cl.evt.gz.gpg
80906801X/xti/event_cl/ni80906801X_0mpu7_ufa.evt.gz.gpg

80906801X/xti/event_uf/ni80906801X_0mpu0_uf.evt.gz.gpg
80906801X/xti/event_uf/ni80906801X_0mpu1_uf.evt.gz.gpg
80906801X/xti/event_uf/ni80906801X_0mpu2_uf.evt.gz.gpg
80906801X/xti/event_uf/ni80906801X_0mpu3_uf.evt.gz.gpg
80906801X/xti/event_uf/ni80906801X_0mpu4_uf.evt.gz.gpg
80906801X/xti/event_uf/ni80906801X_0mpu5_uf.evt.gz.gpg
80906801X/xti/event_uf/ni80906801X_0mpu6_uf.evt.gz.gpg

80906801X/xti/hk/ni80906801X_0mpu0.hk.gz
80906801X/xti/hk/ni80906801X_0mpu1.hkgz
80906801X/xti/hk/ni80906801X_0mpu2.hk.gz
80906801X/xti/hk/ni80906801X_0mpu3.hk.gz
80906801X/xti/hk/ni80906801X_0mpu4.hk.gz
80906801X/xti/hk/ni80906801X_0mpu5.hk.gz
80906801X/xti/hk/ni80906801X_0mpu6.hk.gz
80906801X/xti/hk/ni80906801X_0mpu0.conf.gz
80906801X/xti/hk/ni80906801X_0mpu1.conf.gz
80906801X/xti/hk/ni80906801X_0mpu2.conf.gz
80906801X/xti/hk/ni80906801X_0mpu3.conf.gz
80906801X/xti/hk/ni80906801X_0mpu4.conf.gz
80906801X/xti/hk/ni80906801X_0mpu5.conf.gz
80906801X/xti/hk/ni80906801X_0mpu6.conf.gz

80906801X/xti/products/ni80906801Xmpu7_sr.lc.gz.gpg
80906801X/xti/products/ni80906801Xmpu7_sr.pha.gz.gpg
80906801X/xis/products/ni80906801Xmpu7.rps.gz.gpg
80906801X/xis/products/ni80906801X_lc.gif.gz.gpg
80906801X/xis/products/ni80906801X_pi.gif.gz.gpg
80906801X/xis/products/ni80906801Xmpu7_bg.lc.gz.gpg
80906801X/xis/products/ni80906801Xmpu7_bg.pha.gz.gpg
```

Listed in file *nihkgen1_filelist.txt* and *nihkgen2_filelist.txt*:

```
ni80906801X.hk.gz
```

Listed in file *nihkmpu1_filelist.txt* and *nihkmpu2_filelist.txt*:

```
ni80906801X_0mpu0.hk.gz
ni80906801X_0mpu1.hk.gz
```

.

```
ni80906801X_0mpu2.hk.gz
ni80906801X_0mpu3.hk.gz
ni80906801X_0mpu4.hk.gz
ni80906801X_0mpu5.hk.gz
ni80906801X_0mpu6.hk.gz
```

Listed in file *nihkconf1_filelist.txt* and *nihkconf2_filelist.txt*:

```
ni80906801X_0mpu0.conf.gz
ni80906801X_0mpu1.conf.gz
ni80906801X_0mpu2.conf.gz
ni80906801X_0mpu3.conf.gz
ni80906801X_0mpu4.conf.gz
ni80906801X_0mpu5.conf.gz
ni80906801X_0mpu6.conf.gz
```

Listed in file *nidbtable_filelist.txt*:

```
nicermaster.tdat
```

4.2 DTS Commands (NICERMOC to NICERHEA)

a) Non Validated

```
cd notvalidated/nidata1/
dts -noclobber -i dts$(\date +%H%M%S_%y%m%d)nidata1 -se NICERHEA -split -t nidata1 -f nidata1_filelist.txt

cd notvalidated/nihkgen1/
dts -noclobber -i dts$(\date+%H%M%S_%y%m%d)nihkgen1 –se NICERHEA -split -t nihkgen1 -f nihkgen1_filelist.txt

cd notvalidated/nihkmpu1/
dts-noclobber-i dts$(\date+%H%M%S_%y%m%d)nihkmpu1 –se NICERHEA -split -t nihkmpu1 -f nihkmpu1_filelist.txt

cd notvalidated/nihkconf1/
dts -noclobber -i dts$(\date +%H%M%S_%y%m%d)nihkconf1 –se NICERHEA -split -t nihkconf1 -f nihkconf1_filelist.txt
```

b) Validated

```
cd validated/nidata2/
dts -noclobber -i dts$(\date +%H%M%S_%y%m%d)nidata2 -se NICERHEA -split -t nidata2 -f nidata2_filelist.txt

cd notvalidated/nihkgen2/
dts -noclobber -i dts$(\date+%H%M%S_%y%m%d)nihkgen2 –se NICERHEA -split -t nihkgen2 -f nihkgen2_filelist.txt

cd notvalidated/nihkmpu2/
```

dts-noclobber-i dts$(\date+%H%M%S_%y%m%d)nihkmpu2 –se NICERHEA -split -t nihkmpu2 -f nihkmpu2_filelist.txt

cd notvalidated/nihkconf2/
dts -noclobber -i dts$(\date +%H%M%S_%y%m%d)nihkconf2 –se NICERHEA -split -t nihkconf2 -f nihkconf2_filelist.txt

cd notvalidated/nidbtable/
dts –noclobber -i dts$(\date +%H%M%S_%y%m%d)nidbtable –se NICERHEA –split -t nidbtable -f nidbtable_filelist.txt

where
$(date +%H%M%S_%y%m%d)
is the current hour, minute, second, year, month, day, set byparsing the unix ``date'' command.

This creates a directory named, for example, dts210244_150817nidataNICERHEA (August 17, 2015, at 21:02:44), in the staging directory.

.